PR #46706 完整报告

vllm-project/vllm

Remove grok model arch from vllm

合并时间：2026-06-26 14:02

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46706

执行摘要

- 一句话：删除 Grok 模型架构及全部配套代码
- 推荐动作：该 PR 是一个典型的移除过时功能的清理操作，展示了如何系统性地删除一个模型架构及其配套组件。对于仓库维护者，建议关注类似的低使用率模块清理。对于其他开发者，可学习此类 PR 的完整覆盖模式（核心实现、tokenizer、renderer、注册表、配置、测试、文档）。

功能与动机

PR 描述指出，根据 vLLM 内部使用跟踪，Grok 模型使用率极低（过去 3 个月不足 10 小时）。为减少代码维护负担，移除这些过时模型。经与团队确认，相关模型已不再维护。

实现拆解

1. 删除核心模型文件vllm/model_executor/models/grok1.py（-792 行）：移除了 Grok1ForCausalLM 和 Grok1ModelForCausalLM 的完整实现，包括模型配置解析、注意力、MoE 路由、MLP 层（Grok1MLP）、RoPE 参数解析等。
2. 删除自定义 tokenizervllm/tokenizers/grok2.py（-452 行）：移除了基于 tiktoken 的 Grok2Tokenizer，及其配套的配置加载、编码支持和特殊 token 管理。
3. 删除自定义 renderervllm/renderers/grok2.py（-90 行）：移除了 Grok2Renderer，它负责消息渲染和聊天模板处理。
4. 删除测试tests/models/language/generation/test_grok.py（-43 行）：移除了 Grok2 的 dummy 生成测试，以及 tests/tokenizers_/test_basic.py 和 tests/models/registry.py 中与 Grok 相关的引用。
5. 更新注册表和配置：
   - 在 vllm/model_executor/models/registry.py 中，从 _VLLM_MODEL_ARCHITECTURES 移除 Grok1 两条映射，并加入 _NON_SUPPORTED_ARCHITECTURES。
   - 在 vllm/model_executor/models/transformers/moe.py 中，移除 Grok1 独有的 expert weight 映射（"linear", "linear_1", "linear_v"）和 gelu 激活选择。
   - 在 vllm/config/model.py 中移除对 Grok 架构的特殊处理。
   - 在 vllm/tokenizers/registry.py 和 vllm/renderers/registry.py 中移除 Grok2 注册条目。
6. 更新文档docs/models/supported_models.md 中删除 Grok 相关行。

关键文件：

- vllm/model_executor/models/grok1.py（模块 模型层；类别 source；类型 deletion；符号 _get_num_experts, _get_moe_intermediate_size, _get_grok_version, _get_rope_parameters）：核心模型架构文件，删除 Grok1/Grok2 全部模型实现（792 行），包含模型类、MoE 配置、RoPE 参数解析、MLP 层等。
- vllm/tokenizers/grok2.py（模块 分词器；类别 source；类型 deletion；符号 _maybe_load_tokenizer_config, _load_tiktoken_encoding, encode_patched, Grok2Tokenizer）：自定义 Grok2 tokenizer，基于 tiktoken 实现，移除自包含的 tokenizer 实现（452 行）。
- vllm/renderers/grok2.py（模块 渲染器；类别 source；类型 deletion；符号 Grok2Renderer, init, _apply_chat_template, render_messages）：自定义 Grok2 renderer，处理消息渲染和聊天模板，移除 90 行代码。
- tests/models/language/generation/test_grok.py（模块 测试；类别 test；类型 deletion；符号 _grok2_dummy_overrides, test_dummy_generate）：删除 Grok2 的 dummy 生成测试（43 行）。
- vllm/model_executor/models/registry.py（模块 模型注册；类别 source；类型 data-contract）：更新模型注册表：从支持的架构中移除 Grok1 两条映射，并将它们加入 _NON_SUPPORTED_ARCHITECTURES。
- vllm/model_executor/models/transformers/moe.py（模块 MoE 层；类别 source；类型 data-contract）：移除 Grok1 独有的 MoE weight mapping 和激活函数特化。
- vllm/config/model.py（模块 配置；类别 source；类型 data-contract）：移除对 Grok 架构的特殊处理。
- vllm/renderers/registry.py（模块 渲染器注册；类别 source；类型 core-logic）：移除 Grok2 renderer 注册。
- vllm/tokenizers/registry.py（模块 分词器注册；类别 source；类型 core-logic）：移除 Grok2 tokenizer 注册。
- tests/tokenizers_/test_basic.py（模块 测试；类别 test；类型 test-coverage）：移除对 Grok2 tokenizer 的测试引用。
- tests/models/registry.py（模块 测试；类别 test；类型 test-coverage）：移除 Grok 模型在测试注册表中的条目。
- docs/models/supported_models.md（模块 文档；类别 docs；类型 documentation）：更新文档，移除 Grok 模型支持列表。

关键符号：_get_num_experts, _get_moe_intermediate_size, _get_grok_version, _get_rope_parameters, _get_moe_renormalize, Grok1MLP.init, Grok1MLP.forward, _maybe_load_tokenizer_config, _load_tiktoken_encoding, encode_patched, Grok2Tokenizer.init, Grok2Tokenizer.from_pretrained, Grok2Renderer.init, Grok2Renderer.render_messages, Grok2Renderer.render_messages_async, test_dummy_generate

关键源码片段

vllm/model_executor/models/grok1.py

核心模型架构文件，删除 Grok1/Grok2 全部模型实现（792 行），包含模型类、MoE 配置、RoPE 参数解析、MLP 层等。

```
# ---------- 从 vllm/model_executor/models/grok1.py 中移除的代码 ----------
# 以下函数和类用于支持 Grok1 / Grok2 模型的推理，现已删除。

def _get_grok_version(config) -> str:
    """根据 HuggingFace config 判断当前是 Grok1 还是 Grok2"""
    # Grok2 具有 residual_moe 或 moe_intermediate_size 属性
    if getattr(config, "residual_moe", False) or hasattr(config, "moe_intermediate_size"):
        return "grok2"
    return "grok1" # 默认为 Grok1

def _get_num_experts(config) -> int:
    """从配置中获取专家数量，支持 Grok1 和 Grok2 的不同字段名"""
    return getattr(config, "num_experts", getattr(config, "num_local_experts", 8))

class Grok1MLP(nn.Module):
    """Grok1 模型的自定义 MLP 层，使用 GeLU 激活（不同于现代模型的 SwiGLU）"""
    def __init__(self, config, quant_config=None, prefix=""):
        super().__init__()
        # 将 gate 和 up 投影合并为一个 MergedColumnParallelLinear 以优化通信
        self.gate_up_proj = MergedColumnParallelLinear(
            config.hidden_size,
            [config.intermediate_size // 2] * 2,
            bias=False,
            quant_config=quant_config,
            prefix=f"{prefix}.gate_up_proj",
        )
        self.down_proj = RowParallelLinear(
            config.intermediate_size // 2,
            config.hidden_size,
            bias=False,
            quant_config=quant_config,
            prefix=f"{prefix}.down_proj",
        )

    def forward(self, hidden_states):
        gate_up, _ = self.gate_up_proj(hidden_states)
        gate, up = gate_up.chunk(2, dim=-1)
        return self.down_proj(F.gelu(gate) * up)
```

vllm/model_executor/models/registry.py

更新模型注册表：从支持的架构中移除 Grok1 两条映射，并将它们加入 _NON_SUPPORTED_ARCHITECTURES。

```
# vllm/model_executor/models/registry.py 中的变更：
# 在 _VLLM_MODEL_ARCHITECTURES 中移除了两行：
```

```
# - "Grok1ModelForCausalLM": ("grok1", "GrokForCausalLM"),
# - "Grok1ForCausalLM": ("grok1", "GrokForCausalLM"),
# 同时将这些架构加入 _NON_SUPPORTED_ARCHITECTURES:
_NON_SUPPORTED_ARCHITECTURES = {
    "BaichuanForCausalLM": "0.23.0",
    "AquilaModel": "0.24.0",
    "AquilaForCausalLM": "0.24.0",
    "Grok1ModelForCausalLM": "0.24.0", # 新增，标记为不再支持
    "Grok1ForCausalLM": "0.24.0", # 新增，标记为不再支持
}
```

评论区精华

主要讨论围绕删除的合理性展开：

- DarkLight1337 询问是否已获得模型供应商的确认。
- xianbaoqian 回应已与 youkaichao 确认，Grok 1/2 是很旧的模型，团队重组后可能不再维护。
- youkaichao 批准了该 PR。 无其他争议或设计讨论。
- 确认删除权限 (question)：获得内部确认，可以移除。

风险与影响

- 风险：主要风险是如果仍有用户依赖 Grok 模型，移除后他们将无法加载这些模型。但根据使用数据（过去 3 个月不足 10 小时）和团队确认（模型方已不维护），风险极低。技术风险包括遗漏引用，但 PR 系统地删除了所有注册表入口、tokenizer/renderer 注册、配置特化以及文档引用，覆盖全面。两个模型架构都使用 trust_remote_code 且无 transformers 后端 fallback，因此移除后完全不可用。未发现遗留的 import 或配置路径。
- 影响：用户影响：对使用 Grok1/Grok2 的用户是 breaking change，但使用率极低。系统影响：vLLM 代码库减少约 1400 行代码，降低维护负担。团队影响：简化 model_executor 和 tokenizers 目录，避免为过期模型进行兼容性维护。
- 风险标记：兼容性破坏 , 使用率极低

关联脉络

- PR #46071 [Docs] Remove BambaForCausalLM from supported hybrid models list: 类似的清理操作，从文档中移除已删除的模型引用，减少技术债务。
- PR #46405 [Refactor] Remove dead kernel code: 类似的清理操作，移除死内核代码，减少技术债务。